Description Logic as core machinery for the automatic generation of multilingual technical documents

Thorsten Liebig and **Dietmar Rösner**
Otto-von-Guericke-Universität Magdeburg
Institut für Informations- und Kommunikationssysteme
P.O. Box 41 20, D-39016 Magdeburg, Germany
{tliebig|roesner}@iik.cs.uni-magdeburg.de

Abstract

Our project aims at the automatic generation of multilingual text for product maintenance and documentation from a structured knowledge representation. The language independent representation is also designed to allow for the qualitative simulation of the documentation steps. Our experience is that a representation system only supporting description logic is not able to master all the requirements that come along with the multiple usage of the knowledge to be represented. For an application with a dynamically changing knowledge base of definitions, rules and facts additional paradigms like data-driven programming and object-oriented programming are needed. To verify this, we discuss those aspects of our representation which play key roles, in order to support the demands of the domain representation and the requirements of the text generation.

1 Introduction

The availability of technical documentation for consumer goods and machinery in multiple languages is becoming increasingly important, not merely in the context of the European common market. The TECHDOC project[1] [Stede and Rösner, 1994] aims to automatically generate at least parts of multilingual technical documentation from a language independent representation of form and content. The generation is based on language and product specific knowledge, as well as general technical domain knowlege, represented in a description logic. The knowledge has been aquired through the comparative analysis of manuals in different languages [Rösner *et al.*, 1996].

[1]The TECHDOC project is in progess at the FAW Ulm (Research Institute for Applied Knowledge Processing, Ulm, Germany) since mid 1991. Most of the work is now continued at the "knowledge-based systems and document processing group" at the University of Magdeburg, Germany.

The underlying knowledge structure of instruction and maintenance texts can be well formalized by means of *plans* [Rösner *et al.*, 1996]. These plans consist of one or more *actions* or other plans. Actions are characterized by *pre-* and *postconditions.* These conditions interact with the underlying actual status of the knowledge base. Therefore, the availability of adequate and expressive actions depends directly on the chosen model for objects, qualities and states.

1.1 Representation Requirements

A detailed and expressive representation of actions allows for the *qualitative simulation* of the instruction steps, in order to check for completeness and applicability of a plan, or to derive hints and warnings automatically. Consider, for example, the consequences of the instruction for toasting bread using a household toaster: parts of the toaster become extremely hot and/or are energized. With the general background knowledge that the heating wire is a hot object which is not completely shielded from external access, the reasoning system classifies the current situation as potentially dangerous and prompts the generator to create a warning. This example illustrates that for those kinds of reasoning tasks, one will need a domain representation based on a model of physical structuring as well as on dynamically changing qualities and status properties.

To be able to support both — the demands of the domain representation (cf. above) and the requirements of text generation — we make use of a knowledge representation on three levels.

The language dependent knowledge is organized on the topmost level. Our prototypical software system uses the sentence generator PENMAN [Mann, 1983] to produce natural language output[2] from a semantic specification. PENMAN's domain and task independent ontology, called the *upper model* [Bateman *et al.*, 1990], contains knowledge about *objects*, *processes* and *qualities* on a very

[2]Currently in English, German and French

abstract level. Below such a general, linguistically motivated, classification we have been modelling a level which we call the ***middle model.*** This level contains knowledge about physical and functional objects (e. g. electrical objects, connections etc.) and their potential states (e. g. power-supplied, tightened etc.) in the domain of technical products. Additionally, this level includes the principles which describe how these states can be achieved and how they influence other objects which are related together to form a complex whole. Finally, the *domain model* represents the lowest level and contains knowledge about particular technical devices in a concrete application (e. g. the toaster domain).

1.2 Representation Language

Our experience is that a representation system only supporting description logics is not able to master all the requirements above. The integration of additional programming paradigms in the representation system is therefore a necessary condition, in order to realize adequate models. In our opinion a reasonable representation system must supply an inference engine which successfully integrates frame–like knowledge, as well as rule–like knowledge combined with a generalization of the conception of object–oriented method dispatching.

We are using the knowledge representation tool called LOOM [MacGregor and Bates, 1987], a descendent of the KL-ONE family and based on LISP. LOOM is a good candidate for our representation purpose because it offers more than just a very powerful description logic. Besides this LOOM is already used (although not really exploited) for the *upper model* of PENMAN. In addition LOOM provides powerful deductive support with forward– and backward–chaining, including both strict and default reasoning and automatic consistency checking. It also offers procedural programming, a full first–order query language, production rules, multiple knowledge bases, and object-oriented methods [MacGregor, 1991b].

In addition, the empirical analysis of six terminological representation systems in [Heinsohn *et al.*, 1994] showed LOOM to be the most expressive and fastest one.[3] However, LOOM's inference algorithm is incomplete [MacGregor, 1991a]. This deliberate conceptual decision of LOOM's developers is not surprising since it has been shown that determining subsumption between terms, which is needed for the classification process, is NP-hard [Domini *et al.*, 1991] or even undecidable [Schmidt-Schauß, 1989] for reasonably expressive languages.

[3]The representation systems were: BACK, CLASSIC, KRIS, LOOM, MESON and SB-ONE.

2 Extending the Modelling Capacities

For any adequate modelling of physical systems, it is important to identify the relevant phenomena, and to identify just the appropriate level of detail to model each phenomenon. Hence, a knowledge engineer must have the a priori information about the inferences to be drawn in his particular application. Adequate models, therefore, incorporate abstractions and approximations that are well suited to the problem solving task [Nayak, 1995] and to the expressiveness, performance and completeness of the representation system. Moreover, the representation should conform to the crucial modelling quality factors of object–oriented systems: reusability, understandability and extensibility [Meyer, 1988].

To identify the relevant phenomena and the appropriate level of detail, we focused our objects from a functional and structural perspective, we will not address here (but cf. [Liebig and Rösner, 1996]).

2.1 Part–Whole Relations

The *part–whole* relation plays a fundamental role in the description of complex objects. This relation can be found in many different domains. As in Franconi's [Franconi, 1993] proposal, we model the part–whole relationship as a reflexive, anti–symmetric and transitive relation $\succeq$:

$$\forall x. \succeq (x,x).$$
$$\forall x,y. \succeq (x,y) \wedge \succeq (y,x) \rightarrow x = y.$$
$$\forall x,y,z. \succeq (x,y) \wedge \succeq (y,z) \rightarrow \succeq (x,z).$$

As in Sattler's [Sattler, 1995] concept language $\mathcal{P}$ for engineering applications, our part–whole relations are *direct* part–whole relations in the sense that they must satisfy the *immediate inferior* definition.[4] As noted in [Artale *et al.*, 1995], this constraint should affect the ABox reasoning process in order to discard non–intended models. In contrast to Sattler's application, which lacks ABox reasoning, we use data–driven rules in order to conform to the above definitions of immediate inferiors and anti–symmetry. These rules observe the actual knowledge base and cause a warning whenever an assertion violates a particular definition.

As discussed in [Artale *et al.*, 1995], part–whole relations cannot simply be modelled by ordinary attributes like price or color. The representation formalism should take their specific meaning and their transitivity rules into account. To illustrate this, consider the example taken from [Artale *et al.*, 1995]: "an arm is part of a musician, the musician is part of an orchestra, but it would sound a bit strange to state that the arm is part of the orchestra". The inacceptability of this inference is due

[4]Given a partially ordered set P, we say that a is an immediate inferior of b if $a < b$ and there does not exist an $x \in P$ such that $a < x < b$.

to a mixing of two different interpretational meanings of the the part–whole relation.

Winston, Chaffin and Herrmann [Winston *et al.*, 1987] proposed a distinction between six kinds of specialized part–whole relations to overcome such problems: *Component/Integral-Object, Member/Collection, Portion/Mass, Stuff/Object, Feature/Activity* and *Place/Area.*

These distinctions allow more suggestive reasoning mechanisms along the part–whole relationships, as far as one single kind of relation is involved.[5] All specific part–whole relationships are modelled as specializations of a general part–whole relation. The transitive version of this relation does, therefore, hold between parts and wholes which have chains of different kinds of part–whole relations between them.

To determine transitive part–whole relationships we have defined additional relations capturing the transitive closure of each of the relations above.[6]

The most important part–whole relationship in our context is between *integral-objects* which have a structure, and their *components* which have a specific functionality and which are separable.

Part–whole relations can be also defined as *essential.* Instances in the range of essential relations are then automatically classified as essential–parts. This allows for the automatic determination of relevant components in order to support the knowledge engineer in building up his domain. Knowledge about essential parts is available at concept level, so even when there are no instances created, the user can ask for the essential parts of a specific device (example in [Liebig and Rösner, 1996]). To continue in this direction, we propose automatic generation of all essential parts when instantiating the respective whole. In order to have a generic function for this purpose, it is necessary to generalize the conception of object–oriented method dispatching to concepts and relations.

2.2 Functional Aspects

The representation we propose combines *structural* and *functional* information about a complex object (e. g. a device). As discussed in [Keuneke, 1991], functional structuring is useful for problem-solving mechanisms, which must often decompose the device's function into the functicion of the components. The functional specification describes the device's goals at a level of abstraction that is of interest at the object level [Keuneke, 1991].

[5]However, the composition of different part–whole relations can have relevant meanings too, but not in all cases (see composition table in [Sattler, 1995]).

[6]The definition of the transitive closure includes the filtering of the reflexivity property of the part–whole relations in order to prevent cyclic paths. For a detailed discussion see [Liebig and Rösner, 1996].

We use a model based on a structural organization enriched with functional components. The function of a device is its intended purpose, which is achieved by behaviours [Keuneke, 1991]. Our model represents behaviour as the causal sequence of transitions of partial states/predicates. Determining the actual states depends on the individual abstract object. An electrical control appliance has, for example, the status "on" if all of its components are in a status in which they close the underlying electrical circuit. The control appliance of electrical devices usually consists of a set of switches. These different kinds of switches (e.g. binary switch, tune etc.) can be adjusted by using a generic action. This action must be applicable to all kind of switches and should therefore use object–oriented method dispatching, generalized to the purposes of description logics.

In order to capture the essentials of complex objects we need to express "vertical" and "horizontal" relationships and constraints.

2.3 Vertical Inheritance

Vertical dependencies can be differentiated into relationships between the *existence* of the whole and the existence of certain parts, and relationships between the *properties* of a whole and the properties of its parts [Simons, 1987].

Existential dependencies have been addressed by proposing relations as essential.

The class of relationships between the properties of a whole and the properties of its parts (and vice versa) can be differentiated into [Artale *et al.*, 1995]:

(a) Properties which the parts inherit from the whole.

(b) Properties which the whole inherits from its parts.

(c) Properties of the parts which are systematically related to properties of the whole (These are not yet captured in our model).

We describe the first two varieties in turn:

(a) The location is a property which parts inherit from the whole (with respect to a certain granularity). In order to inherit properties of this kind we have written macros expressing Franconi's [Franconi, 1993] *left* and *right distributive* quantifiers for relations from his language $\mathcal{ALCS}$:

$$\begin{array}{lll} \lhd\, C.R(a,b) & \text{iff} & \forall x.(\succeq (a,x) \land C(x)) \rightarrow R(x,b) \\ \rhd\, C.R(a,b) & \text{iff} & \forall x.(\succeq (b,x) \land C(x)) \rightarrow R(a,x). \end{array}$$

The operators $\lhd$ and $\rhd$ express the left and right distributive readings. They can be qualified by a qualification predicate C, which was omitted in our macros (formally we assume $C \equiv \top$). These macros extend the description logic by adding backward–chaining implication rules to the knowledge base. Figure 1 shows the inheritance of a property represented by the relation R.

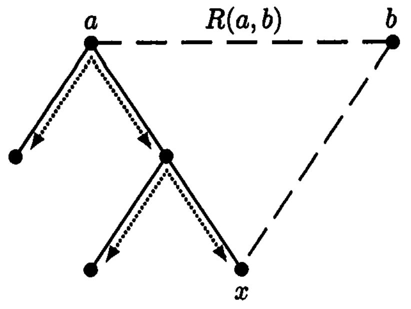

Figure 1: The thin lines indicate the part–whole relations ($\succeq$); the dashed line represents the relation R which is inherited (dotted arrows) left–distributive to all parts of a (displayed for x only).

Since we also express properties in form of concepts, we extend the qualified plural quantifiers with a distributive reading for concepts:

$$\diamond D(a) \quad \text{iff} \quad \forall x. \succeq (a,x). \rightarrow D(x)$$

(b) Properties which the whole inherits from its parts are, in the domain of a household toaster, "on" or "power–supplied". In order to inherit these properties upward, we have written macros which expand to backward–chaining implication rules, similar to Franconi's qualified plural quantifiers, but based on the inverse of the part–whole relation.

2.4 Horizontal Inheritance

Horizontal relationships are composed of constraints among parts which characterize the integrity of the whole. Although they are important for capturing the notion of a whole, they find little attention in current modelling formalisms [Artale *et al.*, 1995].

To determine the energized or hot parts of a device, for example, it is necessary to model the electrical connections of the components of the device. In order to recognize an electrical object as energized, the reasoning mechanism must find those objects which have an electrical connnection to an internal or external power supply. Additionall, the status of any switch on this path must be taken into account. The determination of this path along the electrical connection between objects is directly encoded in LISP as a depth–first search extended with additional filtering procedures.

Horizontal and vertical inheritance often influence each other in order to create complex dependencies. For example, the temperature status of the heating wire depends on the horizontal constraint of being energized. The temperature status of the toaster, therefore, is influenced by the temperature of the heating wire in a vertical manner.

2.5 Temporal Aspects

In a first version of the knowledge base switching the toaster "on" or "off" does immediatly influence the temperature status of the heating wire and the toaster. However, this does not model the real world adequately to be able to derive knowledge about hot objects. For example, the heating wire does not become hot until it has been energized for a little while. Analogously, the wire remains hot for a few minutes after it is no longer energized.

To capture these regularities, one will need a description logic with temporal extensions. LOOM's extension for temporal concepts and relations allows us to make factual assertions about role fillers and instances that hold only over specified intervals, rather than being universally true.

A drawback of the current version of LOOM: it was not possible to express some relevant temporal dependencies in concept or relation definitions directly. We were forced to write production rules in order to detect relevant temporal changes in the knowledge base. The corresponding explicit temporal assertions were placed in the action parts of these rules.

3 Concluding Remarks

As a guideline for structuring complex objects, we propose a model using a physically oriented organization enriched with functional extensions. The transitive part–whole relation and the ability to inherit properties vertically and horizontally play key roles in this structuring.

This skeleton can be used as a basis for a large variety of different modelling purposes, not only in the domain of technical products. Imagine, for example, the modelling of a structured organisation: their administration processes and the inheritance of competences and tasks.

We have enriched the representation language in order to simply enable upwards or downwards inheritance of concepts or relations.

In order to reject unintended models, we use ABox reasoning. For this purpose we needed production rules suitable for triggering concepts and relations.

Object–oriented methods, gerneralized for concepts and relations, were needed for generically applicable creation or adjust methods.

In our opinion, description logic can only build the core of a programming envionment which fulfills the requirements that came along with the multiple uses (text generation, qualitative simulation etc.) of the knowledge to be represented. Additional paradigms like data–driven programming and object–oriented programming are needed. Indeed, theses paradigms have to be carefully integrated into the description language without missing important classes of inferences.

Suggestions for future work include a more conformed integration of temporal extensions into the model. Since many properties are time–dependent, it seems inadequate to write production rules for every particular temporal concept or relation in order to assert temporal facts.

References

[Artale *et al.*, 1995] Alessandro Artale, Enrico Franconi, Nicola Guarino, and Luca Pazzi. Part–whole relations in object-centered systems: An overview. *Data & Knowledge Engineering - North–Holland, Elsevier*, December 1995.

[Bateman *et al.*, 1990] John A. Bateman, Robert T. Kasper, Johanna D. Moore, and Richard A. Whitney. A general organization of knowledge for natural language processing: the penman upper model. Technical report, ISI, 1990.

[Domini *et al.*, 1991] F. Domini, M. Lenzerini, D. Nardi, and W. Nutt. The complexity of concept languages. In J. F. Doyle, R. Files, and Erik Sandewall, editors, *Principles of Knowledge Representation and Reasoning, Proceedings of the Second International Conference (KR '91)*, pages 151 – 162, Cambridge, MA, April 1991. Morgan Kaufmann Publishers, Inc., San Francisco, CA.

[Franconi, 1993] Enrico Franconi. A treatment of plurals and plural quantifications based on a theory of collections. *Minds and Machines*, 3:453 – 474, 1993.

[Heinsohn *et al.*, 1994] Jochen Heinsohn, Daniel Kudenko, Bernhard Nebel, and Hans-Jürgen Profitlich. An empirical analysis of terminological representation systems. *Artificial Intelligence*, 68(2):367 – 398, 1994.

[Keuneke, 1991] Anne M. Keuneke. Device representation, the significance of functional knowledge. *IEEE EXPERT*, pages 22 – 25, April 1991.

[Liebig and Rösner, 1996] Thorsten Liebig and Dietmar Rösner. Modelling of reusable product knowledge in terminological logics: a case study. In *Proceedings of the First International Conference on Practical Aspects of Knowledge Management, — to appear —*, 1996.

[MacGregor and Bates, 1987] R. MacGregor and R. Bates. The LOOM knowledge representation language. Technical report, ISI/RS-87-188, ISI, University of Southern California, 1987.

[MacGregor, 1991a] Robert M. MacGregor. Inside the LOOM Description Classifier. *SIGART Bulletin*, 2(3):88 – 92, June 1991.

[MacGregor, 1991b] Robert M. MacGregor. Using a description classifier to enhance deductive inference. In *Proceedings of the Seventh IEEE Conference on AI Applications*, pages 141 – 147, 1991.

[Mann, 1983] William C. Mann. An overview of the PENMAN text generation system. In *Proceedings of the National Conference on Artificial Intelligence*, pages 261 – 265. AAAI, August 1983.

[Meyer, 1988] B. Meyer. *Object–oriented Software Construction.* Prentice Hall, New York, 1988.

[Nayak, 1995] P. Pandurang Nayak. *Automated Modeling of Physical Systems.* Number 1003 in Lecture Notes in Artificial Intelligence. Springer Verlag, Berlin, 1995.

[Rösner *et al.*, 1996] Dietmar Rösner, Björn Höfling, and Knut Hartmann. From natural language documents to sharable product knowledge. In *Proceedings of the First International Conference on Practical Aspects of Knowledge Management, — to appear —*, 1996.

[Sattler, 1995] Ulrike Sattler. A concept language for an engineering application with part–whole relations. In A. Borigida, M. Lenzerini, D. Nardi, and B. Nebel, editors, *Proccedings of the International Workshop on Description Logics*, pages 119 – 123, Rome, Italy, June 1995.

[Schmidt-Schauß, 1989] M. Schmidt-Schauß. Subsumption in KL-ONE is undecidable. In H. J. Levesque and R. Reiter, editors, *Principles of Knowledge Representation and Reasoning, Proceedings of the First International Conference (KR '89)*, pages 421 – 431, Toronto, ON, May 1989. Morgan Kaufmann Publishers, Inc., San Francisco, CA.

[Simons, 1987] Peter Simons. *Parts: A Study in Ontology.* Clarendon Press, Oxford, 1987.

[Stede and Rösner, 1994] Manfred Stede and Dietmar Rösner. Generating multilingual documents from a knowledge base: The TECHDOC project. In *COLING-94, Proceedings*, Kyoto, 1994.

[Winston *et al.*, 1987] Morton Winston, Roger Chaffin, and Douglas Herrmann. A taxonomy of part-whole relations. *Cognitive Science*, 11:417–444, 1987.